ANL-ART-159

Status of SAS4A/SASSYS Code Developments (FY2018)

Nuclear Science and Engineering Division

DOCUMENT AVAILABILITY

Online Access: U.S. Department of Energy (DOE) reports produced after 1991 and a growing number of pre-1991 documents are available free via DOE's SciTech Connect (http://www.osti.gov/scitech/)

Reports not in digital format may be purchased by the public from the National Technical Information Service (NTIS):
U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Rd
Alexandria, VA 22312
www.ntis.gov
Phone: (800) 553-NTIS (6847) or (703) 605-6000
Fax: (703) 605-6900
Email: orders@ntis.gov

Reports not in digital format are available to DOE and DOE contractors from the Office of Scientific and Technical Information (OSTI):
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831-0062
www.osti.gov
Phone: (865) 576-8401
Fax: (865) 576-5728
Email: reports@osti.gov

ANL-ART-159

Status of SAS4A/SASSYS Code Developments (FY2018)

prepared by
T. H. Fanning, A. J. Brunett, A. Dix, E. Gonzalez, K. Knecht, J. Subick II, E. Wallace

Nuclear Science and Engineering Division
Argonne National Laboratory

September 30, 2018

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Introduction

SAS4A/SASSYS is a simulation tool used to perform deterministic analysis of anticipated events as well as design basis and beyond design basis accidents for advanced liquid-metal-cooled nuclear reactors. [1] With its origin as SAS1A in the late 1960s, the SAS series of codes has been under continuous use and development for nearly fifty years. It has been identified as a critical element of safety analysis capabilities for the U.S. Department of Energy, [2] and it will be used to perform the transient safety analysis required to support the authorization basis for the Versatile Test Reactor (VTR).

Version 5.2.1 of SAS4A/SASSYS was completed in February 2018 and released to users following a software quality assurance review. This is a minor update that addresses a small number of issues that have been identified since the release of Version 5.2 in the prior fiscal year. A second minor update (Version 5.2.2) is expected to be released in the first quarter of FY2019. Version 5.3 is also pending completion, and it will introduce more detailed reactivity feedback models and more flexible input capabilities.

In addition to code updates, improvements have been made to unit testing, regression testing, code coverage analysis, and the verification and validation test suite.

This report summarizes the code development and update activities carried out during FY2018. An overview of the current code status — including implementation of a formal software quality assurance program — is provided in Section 2. Issues addressed in the release of Version 5.2.1 are summarized in Section 3, and Section 4 describes improvements and updates that have been completed for the Version 5.3 release.

2 Current Status

2.1 SQA Plan

The SAS4A/SASSYS Software Quality Assurance Program provides the controls and processes necessary to enable continuous, high-quality software improvement while meeting user and programmatic sponsor requirements. The Software QA Plan (SQAP) delineates the SQA Program framework for SAS4A/SASSYS by describing the Program activities, organization, and documentation, and by clearly defining the interconnection of all Program elements. Program approval and formal implementation of the SAS4A/SASSYS SQAP occurred in March 2018.

The SAS4A/SASSYS SQAP has been derived from the following standards, requirements, and guidance with the aim of supporting both DOE authorization and commercial licensing cases:

- DOE Order 414.1D, *Quality Assurance* [3],
- DOE Guide 414.1-4, *Safety Software Guide for Use with 10 CFR 830 Subpart A, Quality Assurance Requirements, and DOE O 414.1C, Quality Assurance* [4],
- ASME NQA-1-2008 with 2009 Addenda, *Quality Assurance Requirements for Nuclear Facility Applications* [5] [6], and
- IEEE Std 730-2002, *Software Quality Assurance Plans* [7].

2.2 Government Use

SAS4A/SASSYS is used in several U.S. Department of Energy-sponsored research activities. Continued maintenance and improvement of the SAS4A/SASSYS code system is motivated by the importance of its simulation capability to these programs and to domestic and international collaborations. U.S. DOE activities that rely on SAS4A/SASSYS are summarized in Table 1.

Table 1: U.S. DOE Research and Development Activities Using SAS4A/SASSYS

Program/Activity	Description
Versatile Test Reactor	The VTR is a proposed integrated facility for investigations into the behavior of fuels and materials subjected to high, fast neutron fluences prototypic of fast reactor environments. SAS4A/SASSYS will be used for simulating postulated accident sequences in support of DOE authorization for operations.
FFTF Benchmark	The U.S. DOE is preparing benchmark specifications for the Passive Safety Tests (PST) carried out at the Fast Flux Test Facility between 1984 and 1986. The most prominent tests were the loss of flow without scram (LOFWOS). In collaboration with PNNL, Argonne is assessing the benchmark specifications and preparing SAS4A/SASSYS models for verification and validation purposes.
DOE/CEA Bilateral Collaboration	An implementation agreement has been established between the U.S. DOE and the Commissariat à l'énergie atomique et aux énergies alternatives (CEA) of France for cooperation in low carbon energy technologies. One purpose of the agreement is to evaluate the safety performance of advanced sodium fast reactor designs. DOE participates in this collaboration using the SAS4A/SASSYS safety analysis code.
DOE/JAEA Bilateral Collaboration	The Civil Nuclear Energy Research and Development Working Group (CNWG) was established by the U.S.-Japan Bilateral Commission on Civil Nuclear Cooperation in 2012 to enhance coordination of joint nuclear research and development. The Japan Atomic Energy Agency (JAEA) and Argonne are collaborating under the CNWG to improve the oxide fuel severe accident modeling capabilities in SAS4A/SASSYS to support JOYO relicensing.
NEUP (University of California—Berkeley)	The University of California at Berkeley is using SAS4A/SASSYS to evaluate safety benefits that might be achieved with autonomous reactivity control devices in sodium-cooled fast reactors.
NEUP (Kansas State University)	Kansas State University is preparing experiments with liquid gallium that can improve the modeling of thermal stratification in SFRs. In collaboration, the University of Illinois will develop improved stratification models that could be incorporated into SAS4A/SASSYS.
NEUP (University of Wisconsin)	The University of Wisconsin is preparing experiments with liquid sodium that can improve the modeling of thermal stratification in SFRs. In collaboration, Virginia Commonwealth University is working to develop improved models that could be incorporated into SAS4A/SASSYS.

A significant development in this area is the formation of the Versatile Test Reactor (VTR) Program. The VTR is a proposed integrated facility for investigations into the behavior of fuels and materials subjected to high, fast neutron fluences prototypic of fast reactor environments. Detailed design of the reactor core is currently being performed. The current design is a 300 MW sodium-cooled pool-type fast reactor fueled with ternary U-20Pu-10Zr metallic fuel. SAS4A/SASSYS will be used for simulating postulated accident sequences in support of DOE authorization for operations.

2.3 New License Agreements

During FY2018, six new license agreements were established for SAS4A/SASSYS. These are summarized in Table 2. Globally, there are approximately thirty licensed users/organizations with access to SAS4A/SASSYS or Mini SAS.

Table 2: SAS License Agreements Established during FY2017

Organization	Code	Purpose
Westinghouse Electric Company	SAS4A/SASSYS	LFR Concept Analyses
Fauske and Associates, LLC	SAS4A/SASSYS	GAIN Award Studies
Purdue University	Mini SAS	Senior Design Studies
Applied Programming Technology, Inc.	Mini SAS	SNAP Plugin Development
University of Maryland	Mini SAS	Dynamic PRA Event Analyses
DOE Idaho Operations	SAS4A/SASSYS	VTR Program Oversight

3 Latest Release

The latest release of SAS4A/SASSYS is Version 5.2.1, which was approved for release in February 2018 and subsequently distributed to licensed users. Version 5.2.1 is a “patch” (i.e. bug-fix) release that addresses the following issues:

Bug Fixes

- `RESTART` files are not generated if `NSTEP = 0`, consistent with documentation.
- Eliminated a potential divide-by-zero error when the Young's Modulus for cladding and fuel are not provided in input.
- Corrected the declaration of a local variable used in debug print statements.
- Corrected an issue where input parameter `IPRION` was not treated consistently when PRIMAR-4 was not being used.
- Eliminated a potential floating-point exception on Windows when debug prints are enabled.
- Eliminated a potential divide-by-zero error when predicting the time-step cutback in PRIMAR-4 if core channel flow rates are not changing.
- Corrected an issue where a non-zero value for `IFLOW` would impact PRIMAR-4 calculations.
- Eliminated a rare divide-by-zero error on Windows caused by poor CPU timing resolution.

Code Changes

- Removed unneeded compilation of `fpoptimizer` from the FParser library.
- Added support for "developer" build to simplify regression testing.

- Updated copyright.
- Replaced non-standard calls to `ETIME` with calls to standard `CPU_TIME`.
- Corrected typos in the README file.

A small number of additional fixes have been implemented since the release of Version 5.2.1. These fixes will be released in the early part of FY2019 as Version 5.2.2.

4 Code Improvements and New Features

Several code development, maintenance, and testing activities are being completed and will be part of a forthcoming release as Version 5.3. These tasks are aimed at improving usability of the software, expanding the testing capabilities for developers, and supporting the verification and validation basis of the code. All activities are being completed under the recently implemented SAS4A/SASSYS SQA Program.

4.1 Code Manual

The SAS4A/SASSYS code manual presently exists as a collection of sixteen large Microsoft Word documents spanning more than 2000 pages. This document format is not amenable to modern software configuration management practices. Many modern, open-source software projects use reStructuredText (reST) markup [8] and Sphinx [9] for generating complex, hyperlinked documentation. reST is a text-based markup format that works exceptionally well with version control and issue tracking systems. Sphinx is used to convert the markup into web- or pdf-based documentation.

An effort to convert the SAS4A/SASSYS code manual from Word to reST format has been nearly completed. Preserving the integrity of the thousands of equations that are documented in the manual was a priority during the conversion. Although most of the equations could be converted automatically, many required manual typesetting and all needed to be verified. Equations are now typeset in the LaTeX [10] format and are rendered in web browsers by the MathJax plug-in [11] developed by the American Mathematical Society (AMS). An example of the new code manual format is shown in Figure 1.

4.2 Automated Code Coverage Metrics

Like many modern compilers, the Intel Fortran compiler has the ability to create instrumented executables that provide valuable runtime statistics, including, but not limited to, coverage of lines, functions/subroutines, or blocks utilized during execution of the application. These statistics are reported by the Intel compiler via either a plain text-, html-, or xml-based report of coverage per file.

The SAS4A/SASSYS source code is organized by module, where a series of source files responsible for calculation of similar phenomena or features are located in a single directory (or module). For example, the majority of source files related to modeling of system thermal hydraulics are located in the PMR4 module, all source files related to main program flow are located in the MAIN module, etc.

While the Intel Fortran code coverage statistics tool is useful for reporting coverage per file (Figure 2 and Figure 3), it is more useful for SAS4A/SASSYS developers to review code coverage per module, as a filename's association with a module is not always obvious. This way, gaps in testing coverage can be identified and resolved on a prioritized basis. Previously, determining code coverage per module required manual grouping of data for each file into its associated module. This process has now been automated.

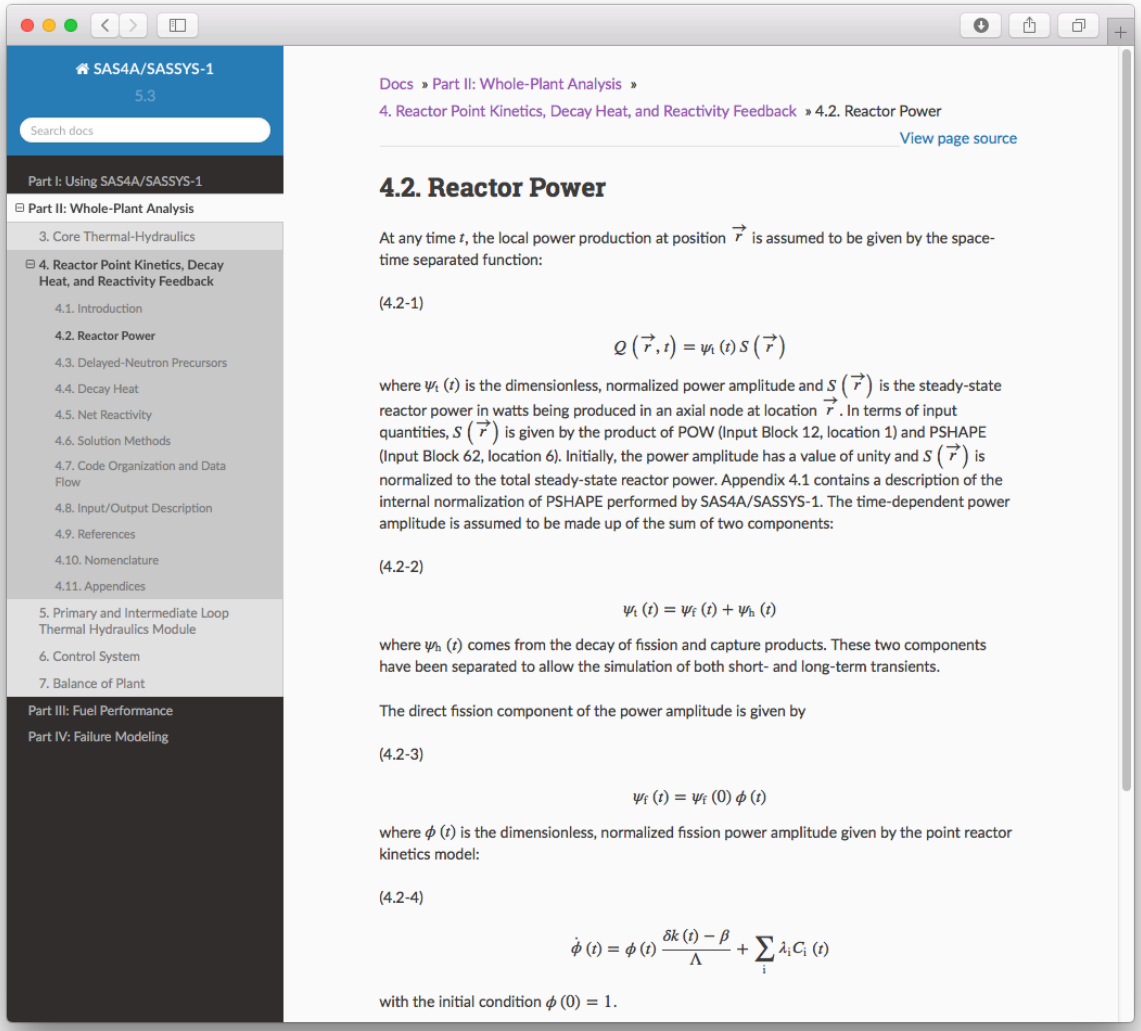


Figure 1: Example of Sphinx-Generated Documentation for SAS4A/SASSYS.

A Python script was developed that parses the text-based code coverage report to compile module-wide statistics and imports the data into an Excel file (Figure 4 and Figure 5). The script is able to dynamically determine filenames and modules. This precludes the need for modification of the script if new source files or modules are added to the source code. Compiled data is exported to an Excel file, with the exported data color coded as per coverage percentage. All data provided in the html report (percent and number of functions and blocks covered, Figure 3) is retained in the Excel-generated report. Some limited error checking and messaging has been integrated into the Python script to help increase usability, ensure meaningful data is produced by the script, and to prevent errant overwrites of data. Several user options were implemented via command-line flags, such the ability to name the exported file, an option to overwrite existing data, and an option to export a series of reports (Figure 6). A simple GUI application (Figure 7) has also been created which maintains the same functionality as the script.

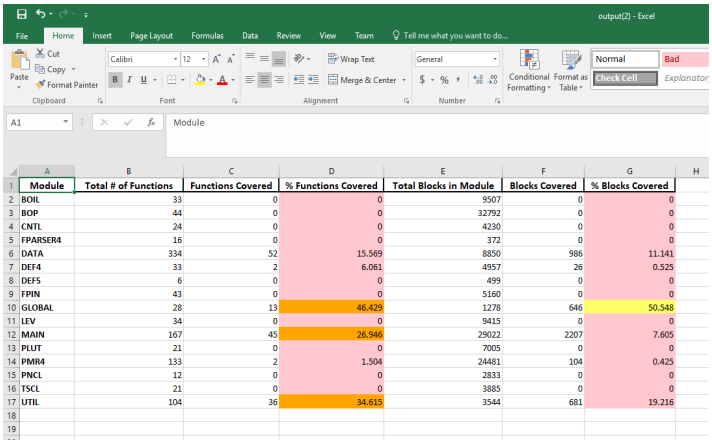
```
file - "/Users/abrunett/Documents/SAS/source/trunk/data/source/INPT_PRIMIN.F90"
function - inpt_primin_mp_inpt_priminclear_ 0   40  0    70.00
function - inpt_primin_mp_inpt_priminreaddat_   0   9   0     0.00
function - inpt_primin_mp_inpt_priminreadinp_   1   8   6    75.00
1   98  12              1
1   103 2               1
1   103 22              0
1   105 2               1
1   105 29              0
1   107 7               1
1   109 2               1
1   111 1               1
function - inpt_primin_mp_inpt_priminreadnml_   0   9   0     0.00
function - inpt_primin_mp_inpt_priminwritedat_  0   7   0     0.00
function - inpt_primin_mp_inpt_priminwriteinp_  0   6   0     0.00
function - inpt_primin_mp_inpt_priminwritenml_  0   7   0     0.00
```

Figure 2: Example of text-based coverage report for a single file.

Covered Files in Test07.26.2017

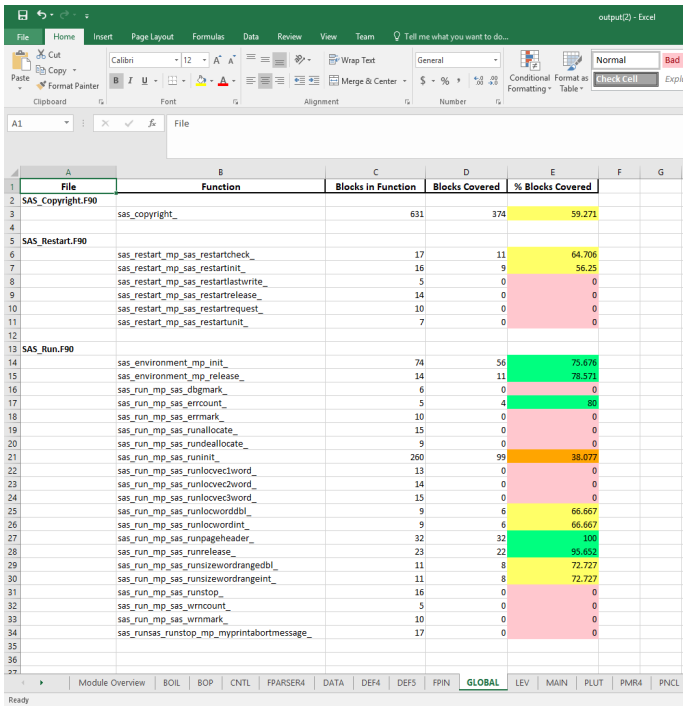
Name	Functions			Blocks		
	total	cvrd	cvrg%	total	cvrd	cvrg%
ANETMP.F90	1	1	100.00	208	67	32.21
AVGVAL.F90	1	1	100.00	53	24	45.28
AVRGIT.F90	1	1	100.00	43	20	46.51
BLKPRN.F90	1	1	100.00	13	13	100.00
BYPSTM.F90	1	1	100.00	114	39	34.21
BYSASA.F90	1	1	100.00	74	8	10.81
CAVITR.F90	2	2	100.00	139	99	71.22
CAVITY.F90	1	1	100.00	118	92	77.97
CAVMOT.F90	1	1	100.00	64	46	71.88
CCLAD.F90	1	1	100.00	7	7	100.00

Figure 3: Example of html-reported coverage statistics.



Module	Total # of Functions	Functions Covered	% Functions Covered	Total Blocks in Module	Blocks Covered	% Blocks Covered
BOIL	33	0	0	9507	0	0
BOP	44	0	0	32792	0	0
CNTL	24	0	0	4230	0	0
FPARSER4	16	0	0	372	0	0
DATA	334	52	15.569	8850	986	11.141
DEF4	33	2	6.061	4957	26	0.525
DEF5	6	0	0	499	0	0
FPIN	43	0	0	5160	0	0
GLOBAL	28	13	46.429	1278	646	50.548
LEV	34	0	0	9415	0	0
MAIN	167	45	26.946	29022	2207	7.605
PLUT	21	0	0	7005	0	0
PMR4	133	2	1.504	24481	104	0.425
PNCL	12	0	0	2833	0	0
TSCL	21	0	0	3885	0	0
UTIL	104	36	34.615	3544	681	19.216

Figure 4: Example Compiled Coverage Statistics (Code-Wide)



File	Function	Blocks in Function	Blocks Covered	% Blocks Covered
SAS_Copyright.F90				
	sas_copyright_	631	374	59.271
SAS_Restart.F90				
	sas_restart_mp_sas_restartcheck_	17	11	64.706
	sas_restart_mp_sas_restartinit_	16	9	56.25
	sas_restart_mp_sas_restartlastwrite_	5	0	0
	sas_restart_mp_sas_restartrelease_	14	0	0
	sas_restart_mp_sas_restartrequest_	10	0	0
	sas_restart_mp_sas_restartunit_	7	0	0
SAS_Run.F90				
	sas_environment_mp_init_	74	56	75.676
	sas_environment_mp_release_	14	11	78.571
	sas_run_mp_sas_dbgmark_	6	0	0
	sas_run_mp_sas_errcount_	5	4	80
	sas_run_mp_sas_errmark_	10	0	0
	sas_run_mp_sas_runallocate_	15	0	0
	sas_run_mp_sas_rundeallocate_	9	0	0
	sas_run_mp_sas_runinit_	260	99	38.077
	sas_run_mp_sas_runlocvec1word_	13	0	0
	sas_run_mp_sas_runlocvec2word_	14	0	0
	sas_run_mp_sas_runlocvec3word_	15	0	0
	sas_run_mp_sas_runlocworddbl_	9	6	66.667
	sas_run_mp_sas_runlocwordint_	9	6	66.667
	sas_run_mp_sas_runpageheader_	32	32	100
	sas_run_mp_sas_runrelease_	23	22	95.652
	sas_run_mp_sas_runsizewordrangedbl_	11	8	72.727
	sas_run_mp_sas_runsizewordrangeint_	11	8	72.727
	sas_run_mp_sas_runstop_	16	0	0
	sas_run_mp_sas_wrncount_	5	0	0
	sas_run_mp_sas_wrnmark_	10	0	0
	sas_runsas_runstop_mp_myprintabortmessage_	17	0	0

Figure 5: Example Compiled Coverage Statistic (per Module)

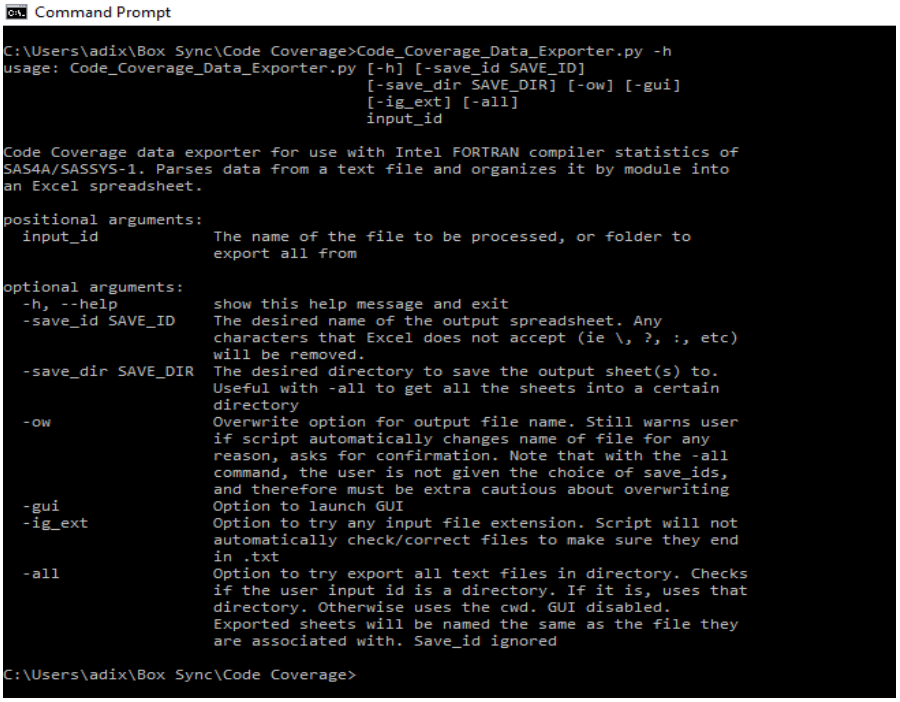


Figure 6: Python Script Command-Line Options

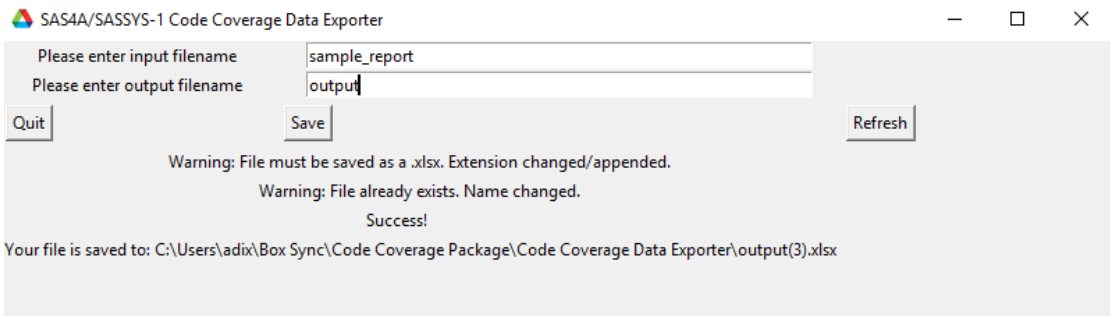


Figure 7: Example GUI Display

4.3 Free-Format Input Processor

The existing SAS4A/SASSYS input processor requires the use of fixed-format input that adheres to code-enforced floating-point, integer, or character type formats. While this data processing methodology provides for robust transfer of input data to code memory, it is generally prone to errors, particularly for new users. In an effort to improve usability and support modernization of the code, the input processor has been updated to permit the use of free-format input.

The update allows free-formatted input lines to contain any number of tab- or space-delimited input records within integer and floating-point data blocks (the processing of character-type data has not been changed). A free-format line is indicated by following the input location with a colon and any number of input records. Comments, indicated by either a pound symbol ("#") or exclamation point ("!"), are permitted within free-format lines, and the existing method for processing of comment lines is maintained. The processing of block identifier records and delimiters remains unchanged, as does the ability to process fixed-format input. An example of permissible free-format input is provided in Figure 8.

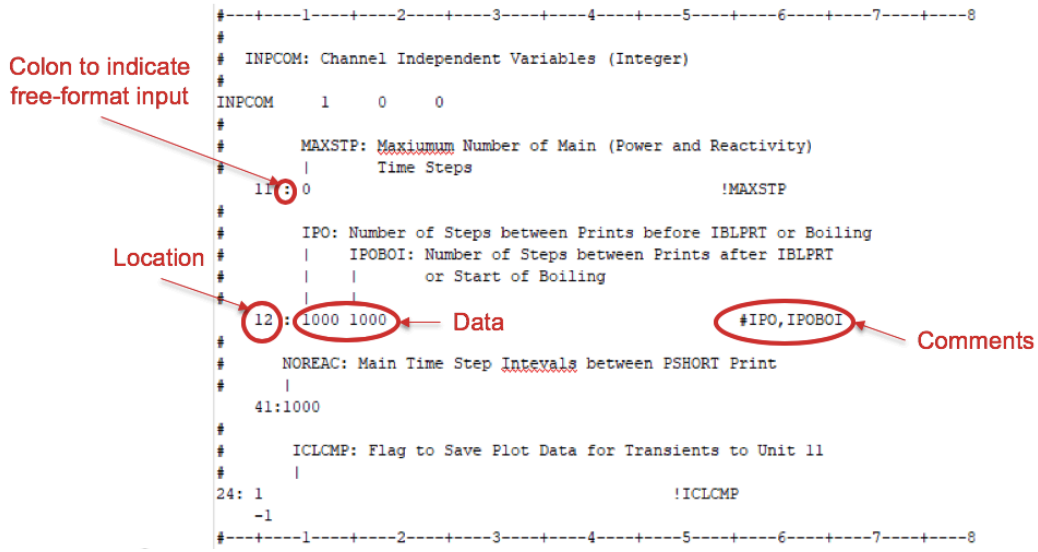


Figure 8: Example of Acceptable Free-Format Input

Robust error checking has been implemented in the updated input processor. Upon detection of an error, code execution is gracefully terminated and the location of error is indicated in the standard output and log files. The code will detect the following error conditions:

- Absence of location preceding a colon;
- Multiple colons in a single line;
- Floating-point data in integer type blocks;
- Non-numeric data where numeric data is expected;
- Absence of data following a colon; and
- Presence of a tab character in a fixed-format input line.

4.4 Formal Review and Integration of Test Problems

The SAS4A/SASSYS Verification and Validation (V&V) Test Suite, which spans simple steady-state and transient phenomena and models, consists of a series of test inputs, analytical solutions (typically spreadsheet), and limited-release reports documenting the test designs and computational and analytical solutions. This collection represents a fundamental basis of verification testing for the code. Creation of the Test Suite and its associated files predates the development and implementation of the SQA Program, and therefore it was not originally subject to the review and acceptance requirements of the Program. Because the Test Suite is expected to be utilized as a foundation for future software qualification and dedication activities undertaken by a user seeking a regulatory license or authorization, it is vital for the Test Suite to undergo review as per Program requirements.

Accordingly, all test problem descriptions, code inputs, analytical solutions, and corresponding results have been reviewed to confirm consistency across all elements. As a result of the review, minor inconsistencies have been identified and corrected to produce a Test Suite with robust supporting documentation that helps to support the code's V&V base. Additional regression test problems which explore the effects of core nodalization schemes were also reviewed as part of this process, and identified inconsistencies have been corrected as needed.

4.5 Buildbot Configuration and Functionality Improvements

A simple Buildbot test system has been utilized for SAS4A/SASSYS maintenance and testing for several years, but recent updates to supported Buildbot protocols and the need for improved acceptance testing have motivated a reconfiguration of the test system. As such, the following changes to the Buildbot system have been implemented:

- The Buildbot configuration has been updated to support the latest version, 1.3;
- Build factories enabling multi-platform support (Linux, Windows, and macOS) have been created, including the ability to check out the source code, compile the source code, check out the V&V Test Suite and accompany testing utility, and execute the tests;
- Repository status monitoring has been created such that changes to specific directories will trigger Buildbot actions; and
- Web-server reporting of worker status for all factories and processes has been configured.

These changes enable a key improvement to acceptance testing. Repository functionality, code acceptability, and test problem status can be efficiently monitored with minimal effort. Gaps in technical review and testing during code and/or test problem merges can now be automatically detected.

4.6 Point Kinetics and Decay Heat Unit Testing

In general, the code architecture of SAS4A/SASSYS does not readily support unit testing. However, the point kinetics and decay heat solvers are implemented in a modular fashion that allows their functionality to be verified through unit testing. Separate unit tests have been established for point kinetics and decay heat, and these modules have been verified against published benchmarks and standards.

The point kinetics solver in SAS4A/SASSYS has been verified against several, highly-accurate benchmark solutions published by Ganapol. [12] SAS4A/SASSYS very accurately predicts power variations in response to step, ramp, and sinusoidal reactivity insertions. Comparisons of the ramp reactivity insertion benchmark are shown in Table 3. As shown in the results, the SAS4A/SASSYS predictions are nearly identical to the published results, with variations due only to numerical round-off errors. Likewise, the decay heat solver in SAS4A/SASSYS has been verified against the published American Nuclear Society standards for decay heat in light water reactors. [13]

Table 3: Normalized Power from Ramp Reactivity Insertion of 0.1 $/s over 10 Seconds

Time(s)	Ganapol	SAS	Error
0	1.000000000E+00	1.000000000E+00	0.00E+00
2	1.338200050E+00	1.338200048E+00	-1.49E-09
4	2.228441897E+00	2.228441889E+00	-3.59E-09
6	5.582052449E+00	5.582052270E+00	-3.21E-08
8	4.278629573E+01	4.278629531E+01	-9.82E-09
10	4.511636239E+05	4.511613959E+05	-4.94E-06
11	1.792213607E+16	1.792213608E+16	5.58E-10

4.7 Improved Stratified Pool Model Output

The SAS4A/SASSYS code has the ability treat liquid stratification in a pool volume due to thermal- and flow-induced development of multiple liquid layers in various stages. Output of this model was provided in the form of limited data prints to the standard output file, with overly detailed data provided in a formatted CSV file at a user-specified frequency. Neither data output structure was conducive to simple plotting for user analyses: the limited set of useful data in standard output would have to be manually extracted by the user, or the user would have to sort through the significant number of extraneous unlabeled debugging data channels to extract useful information from the CSV file to generate plottable data. These aforementioned issues have been corrected to produce a user-friendly binary output file from the stratified pool model.

The following features have been implemented as part of this update:

- Stratified pool model output is now provided in an unformatted data file, and appropriate converting utilities (to XML and CSV formats) have been created which include data labels;
- Converter utilities create plottable data files in which columns for specified data channels are maintained consistently throughout the entire transient, rather than “disappearing” as corresponding coolant layers disappeared (i.e. in the previous configuration, a single column could contain data for layer 1 *or* layer 2); and
- User option to print all stratified pool model data (including debugging prints) or a reduced set of useful transient information.

These changes have been implemented in accordance with modern Fortran coding practices utilizing a modular structure for related subroutines and data structures. These changes will be available in version 5.3 or later of the software.

5 Summary

Over the last several years, the SAS4A/SASSYS safety analysis code has undergone significant revisions to modernize code structure and data management. [14] [15] [16] [17] [18] [19] With the continued importance of advanced, non-LWR reactors, there is also growing interest in applying SAS4A/SASSYS to a number of SFR and LFR concepts. Most significantly, SAS4A/SASSYS will be used to simulate postulated transient accident scenarios to support the authorization basis for the Versatile Test Reactor.

Most of the recent work has focused on improvements to documentation, unit testing, regression testing, code verification, and minor bug fixes. During FY2018, Version 5.2.1 was released, and Version 5.2.2 will be released in early FY2019.

With the growing importance of the VTR program, however, priorities in FY19 will shift to focus on incorporating improvements to the metal fuel performance models. In particular, the module "SSCOMP" will be improved to provide mechanistic models for pre-transient metal fuel characterization, including constituent redistribution, porosity formation, axial fuel growth, and fuel-cladding chemical interactions. These new models will build upon the updated code structure established over the last several years.

6 Acknowledgements

Argonne National Laboratory's work was supported by the U.S. Department of Energy, Office of Nuclear Energy, under contract DE-AC02-06CH11357.

7 Bibliography

[1] T. H. Fanning, A. J. Brunett, T. Sumner and (eds.), *The SAS4A/SASSYS-1 Safety Analysis Code System*, ANL/NE-16/19, Argonne National Laboratory, Nuclear Engineering Division, March 2017.

[2] M. Denman, J. LaChance, T. Sofu, G. Flanagan, R. Wigeland and R. Bari, *Sodium Fast Reactor Safety and Licensing Research Plan — Volume I*, SAND2012-4260, Sandia National Laboratories, May 2012.

[3] U. D. o. Energy, *Quality Assurance*, DOE O 414.1D, 2011.

[4] U. D. o. Energy, *Safety Software Guide for Use with 10 CFR 830 Subpart A, Quality Assurance Requirements, and DOE O 414.1C, Quality Assurance*, DOE G 414.1-4, 2005.

[5] A. S. o. M. Engineers, *Quality Assurance Requirements for Nuclear Facility Applications*, ASME NQA-1-2008, 2008.

[6] A. S. o. M. Engineers, *Addenda to ASME NQA-1-2008: Quality Assurance Requirements for Nuclear Facility Applications*, NQA-1a-2009, 2009.

[7] I. o. E. a. E. Engineers, *IEEE Standard for Software Quality Assurance Plans*, IEEE Std 730-2002, 2002.

[8] "reStructuredText: Markup Syntax and Parser Component of Docutils," 2016. [Online]. Available: http://docutils.sourceforge.net/rst.html. [Accessed 28 June 2018].

[9] G. Brandl, "Sphinx: Python Documentation Generator," [Online]. Available: http://www.sphinx-doc.org/en/master/. [Accessed September 2018].

[10] "LaTeX Documentation," The LaTeX Project, [Online]. Available: https://www.latex-project.org/help/documentation/. [Accessed 28 June 2018].

[11] M. Consortium, "MathJax," 2018. [Online]. Available: https://www.mathjax.org. [Accessed September 2018].

[12] B. D. Ganapol, "A highly accurate algorithm for the solution of the point kinetics equations," *Annals of Nuclear Energy,* vol. 62, p. 564–571, 2013.

[13] A. N. Society, *American National Standard Decay Heat Power in Light Water Reactors*, ANSI/ANS-5.1-2005, American Nuclear Society, 2005.

[14] T. H. Fanning, F. E. Dunn, D. Grabaskas, T. Sumner and J. W. Thomas, *SAS4A/SASSYS-1 Modernization*, ANL-ARC-265, Argonne National Laboratory, September 2013.

[15] T. H. Fanning, A. J. Brunett, T. Sumner and N. Stauff, *SAS4A/SASSYS-1 Modernization and Extension*, ANL-ARC-299, Argonne National Laboratory, September 2014.

[16] T. H. Fanning, A. J. Brunett, T. Sumner and R. Hu, *SAS4A/SASSYS-1 Modernization and Extension*, ANL-ART-30, Argonne National Laboratory, September 2015.

[17] T. H. Fanning, A. J. Brunett and G. Zhang, *SAS4A/SASSYS-1 Code Improvements for FY2016*, ANL-ART-75, Argonne National Laboratory, September 2016.

[18] T. H. Fanning and A. J. Brunett, *Status of the SAS4A/SASSYS-1 Safety Analysis Code*, ANL-ART-97, Argonne National Laboratory, June 2017.

[19] T. H. Fanning, *FY2017 Updates to the SAS4A/SASSYS-1 Safety Analysis Code*, ANL-ART-122, Argonne National Laboratory, September 2017.

Nuclear Science and Engineering Division
Argonne National Laboratory
9700 South Cass Avenue, Bldg. 208
Argonne, IL 60439-4842

www.anl.gov